CSC148 - Linked List Insertion

Our goal for this worksheet is to extend our `LinkedList` class by implementing one of the standard mutating List ADT methods: inserting into a list by index. Here's the docstring of such a method:

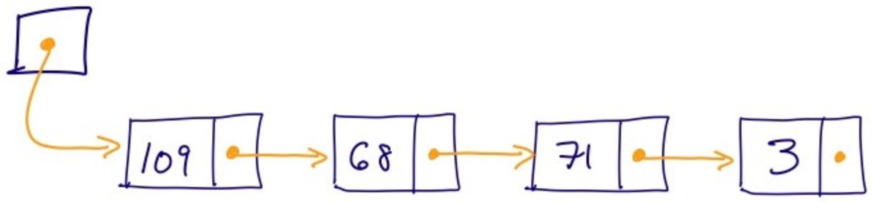
```python
def insert(self, index: int, item: Any) -> None:
    """Insert the given item at the given index in this list.

    Raise IndexError if index > len(self) or index < 0.
    Note that adding to the end of the list is okay.
    """
```

1. Before diving into any code at all, we'll gain some useful intuition by generating some test cases for this method based on two key input properties: the length of the list, and the relationship between `index` and the length of the list. We *won't* care about what item we're inserting (since it could be anything).

   In the table below, we've described some inputs based on these properties. For each row of the table, do the following:

   (a) Draw a linked list of the specified length, using the abstract diagram style shown here. Note that the leftmost box represents the `LinkedList` object itself, with its `_first` attribute referring to the first node in the linked list.



   Note: use a box with a single dot in it to represent an empty list (its `_first` attribute is `None`).

   (b) Show what happens if you insert the number `148` into the list at the given index. You can edit your existing diagram or draw a brand-new one.

Input description	Linked list diagram
`len(self) == 0, index == 0`	
`len(self) == 1, index == 0`	
`len(self) == 1, index == 1`	
`len(self) == 4, index == 0`	
`len(self) == 4, index == 2`	
`len(self) == 4, index == 4`	

2. Using your diagrams as a guide, answer the following questions:

   (a) For what values of `len(self)` and/or `index` would we need to re-assign `self._first` to something new?

   (b) What is the relationship between `len(self)` and `index` that makes `insert` behave the same as `LinkedList.append` from this week's prep?

   (c) In the `len(self) == 4`, `index == 2` case, which *existing* node was actually mutated? Write down the index of this node in the list; hint, it's not the one at index 2!

3. Finally, using these ideas, implement the `insert` method in the space below. Note that you should have two cases: one for when you need to mutate `self._first`, and one where you don't. Also, you'll want to use the same approach as `LinkedList.__getitem__` and keep two parallel variables `curr` and `i`.